Media Data Protection during Execution on Mobile Platforms – A Review

Manoranjan Mohanty, Viktor Do, and Christian Gehrmann
SICS Swedish ICT AB
{manoranjan, viktordo, chrisg}@ sics.se

4 July 2014

SICS Technical Report: T2014:02

Abstract

Multimedia content streaming has become an essential part of digital life. The *media-on-demand* (e.g., video on demand) service of certain enterprises, such as Netflix, Hulu, and Amazon etc. is changing the equations in which media content were accessed. The days, when one has to buy a bulk of media storage devices, or has to wait for the public broadcasting (e.g., television), to enjoy her preferred media has gone. Such change in the way of entertainment, however, has created new issues of piracy and unauthorized media access. To counter these concerns, the digital rights management (DRM) protection schemes have been adopted.

In this report, we investigate one of the most important aspects of the DRM technology: *the problem of protecting the clear text media content when playing licensing protected content on a mobile device*. To this end, we first investigate how this problem has been addressed on different platforms and CPU architectures so far, and then discuss how virtualization technologies can be potentially used to protect the media pipe on mobile platforms. Our study will consider both industry-level and academic-level works, and will discuss the hardware-based and software-based approaches.

Key Words: Digital Rights Management (DRM), ARM TrustZone, Hypervisor, Mobile DRM, Media Pipe Protection.

1 Introduction

The advances in telecommunication and the availability of high-end mobile devices have resulted in a scenario where high quality multimedia data (such as video, audio, and image) is being streamed to anyone, anywhere, and at any time. For this to happen, typically an organization, called as the content provider, hosts a database of multimedia contents, which can be accessed by a user by paying a fee. User's access rights (i.e., licenses), such as allowed time and allowed database-portion, are often determined by the content provider according to the payment. Although such a streaming arrangement can be beneficial both for the content provider (by doing business) and the user (by instantly accessing a media content), issues such as copyright violation and piracy are the major concerns for the content provider. Therefore, the media content must be protected according to the lawful agreement of the content provider.

Media protection technologies in consumer devices started with copyright technologies, which made inroads to DRM technologies [22]. Research in DRM is not new. Besides individual efforts, collaborative organizations, such as Trusted Computing Group (TCG), GlobalPlatform, Open Mobile Alliance (OMA), and Open Mobile Terminal Platform (OMTP), have been formed to jointly provide DRM solutions. Available solutions, however, are still vulnerable, and are not matured for new demands (such as the DRM protection demand for mobile devices [32]).

One of the main issues of the existing DRM schemes is their inability to protect the media pipeline in many general purpose devices (such as mobile devices). Typically, these devices do not support HDCP (High Bandwidth Digital Copy Protection) that helps in hardware-level decryption of protected video. Rather, software-level decryption is used for such low-end devices [27]. For these devices, an adversary, therefore, can get access to the media buffer, and can identify the point of decryption of the encrypted media data by information theoretically analysing three types of data: encrypted data (possessing high randomness and high entropy), clear encoded data (possesses low randomness and high entropy), and clear decoded data (possesses low randomness and low entropy). As a result, the adversary can *steal movies* from the streamed video of most popular video-on-demand providers such as Netflix, Amazon Instant Video, and Hulu etc.

One way to address the above issue is to protect the media pipe by isolating the media rendering operation from the normal OS operations. For example, we can put whole of the media rendering software in the ARM TrustZone (TZ) technology [20], which is now a days available in many mobile devices to protect critical information such as DRM agents, cryptographic keys, and licensing information. Such a solution, however, is not feasible from performance, software architecture, and security perspective, since many domain switches are required, and the risk of compromising the trusted part increases with the increase in the code size. An alternative approach is to take help of hypervisor virtualization, which can isolate operations without taking help of the TrustZone.

Use of virtualization in mobile devices is increasing with the arrival of new mobile hardware support for virtualization [1] [2]. This trend is also evident from the increasing interest for virtualization among embedded systems CPU vendors and the latest CPU-s from ARM with advanced hardware support for virtualization. Virtualization, however, so far has not been widely used to protect DRM data in mobile devices.

In the DRM perspective, virtualization has been mainly used as an alternative to hardware-level security. For example, Berger et al. [28] proposed to virtualize the hardware TPM (Trusted

Platform Module) functionality, so that the remote attestation and sealed storage functionalities can be offered using a virtual machine (VM). With the migration of the VM, the associate virtual TPM (vTPM) is also migrated. Virtualization has also been used to isolate the license related DRM operations from other DRM operations for providing a dependable and scalable DRM license maintenance scheme at the client-side [3]. Recently, Passera et al. [42] used virtualization to perform the DRM operations in a hypervisor. Their scheme can protect the media pipe without using any hardware support to isolate the storage and processing of premium video content from other normal video content. To the best of our knowledge, virtualization, however, has not been extensively used to protect the media pipe in a TrustZone armed mobile device.

We believe that virtualization can be used along with TrustZone to protect the media rendering pipeline.

This document is organized as follows. In Section 2, we provide an overview of DRM technology, and summarize some state-of-the-art DRM schemes. Section 3 discusses how virtualization can be used along with TrustZone, and Section 4 concludes.

2 Related Technology Overview

2.1 DRM overview

Digital rights management (DRM) is a widely used scheme to counter the copyright violation and piracy concerns. In a border sense, DRM is responsible for the management of rights rules (including copyright rules), and their technological enforcement. Typically, DRM (i) allows a content to be associated with a license that specifies the access rights and copyrights, (ii) enforces that the copyright on digital content is respected by checking that the terms stated in the license is being followed by the content user, (iii) allows content providers to add extra (more sophisticated) usage rules, and (iv) supports management of accounting aspects (such as financial aspects) involving the content distribution [30].

A content provider enforces DRM by either cryptographically hiding the multimedia content from an unauthorized user (i.e., who does not hold licenses), or by watermarking the content to trace the users (used to check piracy) [31]. Use of cryptography can prevent the violation of copyright, and is therefore more preferred by a content provider [32].

Figure 1 shows the fundamental structure and function of a typical DRM system. As summarized by Nutzel and Beyer [33], the workflow of the DRM consists of three major components: the content server, the license server, and the DRM client. The content server stores the encrypted content along with metadata such as content ID and the address to the license server. The license server stores the Content Encryption Key (CEK) and usage rights to access CEK. The DRM client, which sits in a user's device, contains a DRM controller, a decoder to play the content, and user's or device's identification mechanism. First, the user gets the content either from the content server or from a third-party source, and tries to render it. From the content metadata, the user's device, however, recognizes that a license is required to render the content, and hence invokes the DRM client. If the license is not locally stored in the user's device, the DRM controller requests the license from the license server by using the license server address. The request includes user's identity, information about the client device (such as device public key), and the content ID from the content package.

After verifying the request parameters, the license generator creates the license containing the CEK and usage rights. The CEK is then encrypted using device public key, and the license including the encrypted CEK is sent to the DRM client. The DRM controller then uses the device private key to decrypt the encrypted CEK. Finally, depending on the usage rights, the content is decrypted by using the CEK, and the clear-text media is rendered in the user's device.

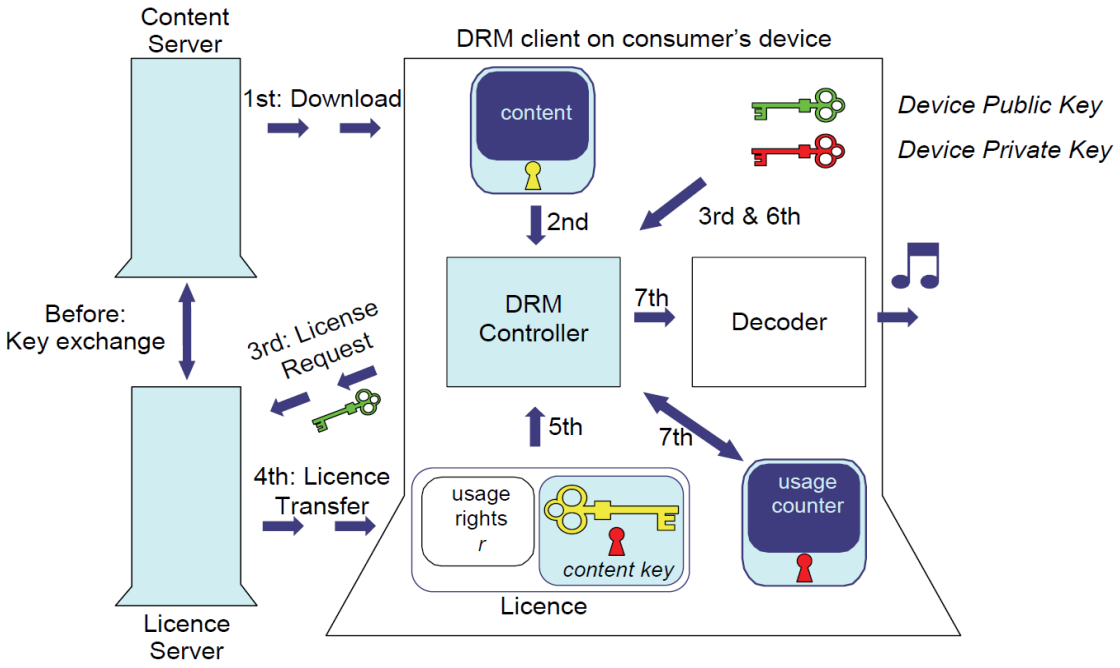

Figure 1: Work flow of DRM [33]

An obvious requirement of a DRM system is that the content server, license server, DRM client, and the communication link between the servers and the client must be secured according to the requirement of the content provider. Securing the servers and the communication links can be easier since they can be controlled by the content provider. Security of the DRM client, however, is a more difficult problem as it is executed on the end user device. In this document, we will focus our discussion on the protection of the DRM client.

In a DRM system, the following assets of the DRM client must be protected [34].

(1) First, DRM's private keys and license (for example, client's device private key), which can be permanently stored in the client's device, must be protected.
(2) Second, DRM's temporary keys, such as the extracted CEK, must be protected.
(3) Third, the decrypted content, which is passed from the DRM client to the codec of media renderer, must be protected.

These DRM assets can be compromised either by attacking the software (e.g., by an attacker having root privileges) or by attacking the hardware (e.g., monitoring the device ports) of the DRM client. To counter such attacks, the following security schemes are typically used [35].

(1) Secure Storage: protecting persistently stored data (e.g. cryptographic keys) that belong to a certain application from being accessed by other applications.

(2) **Isolated Execution:** ensuring that applications execute completely isolated from and unhindered by others and guaranteeing that any code and data is protected at run-time. For example, programs running on the main OS cannot access the code and data memory spaces of the protected execution environment.
(3) **Remote Attestation:** ensuring remote parties that they are dealing with a particular trusted application.
(4) **Secure Provisioning:** sending data to a specific software module of a specific device by protecting data confidentiality and integrity.
(5) **Secure Content Path:** protecting confidentiality, integrity, and authenticity of data communicated between a software module and a peripheral. For example, the channel between the DRM client and the media rendering device (which runs on the main OS) must be protected.

These protection schemes can be either software-based or hardware-based (i.e., combination of hardware and software). Since the hardware-based schemes are more popular and secure than the software-based schemes [34], we will focus our discussion of hardware-based schemes.

The hardware assisted solutions are mainly based on the trusted execution environment (TEE), which is an independent area in the application processor of an electronic device. TEE is separated from the main OS by hardware, and therefore, TEE's CPU and memory are inaccessible to the main OS and applications running over it. ARM TrustZone and Intel SGX (Software Guard Extensions) are examples of some popular TEEs.

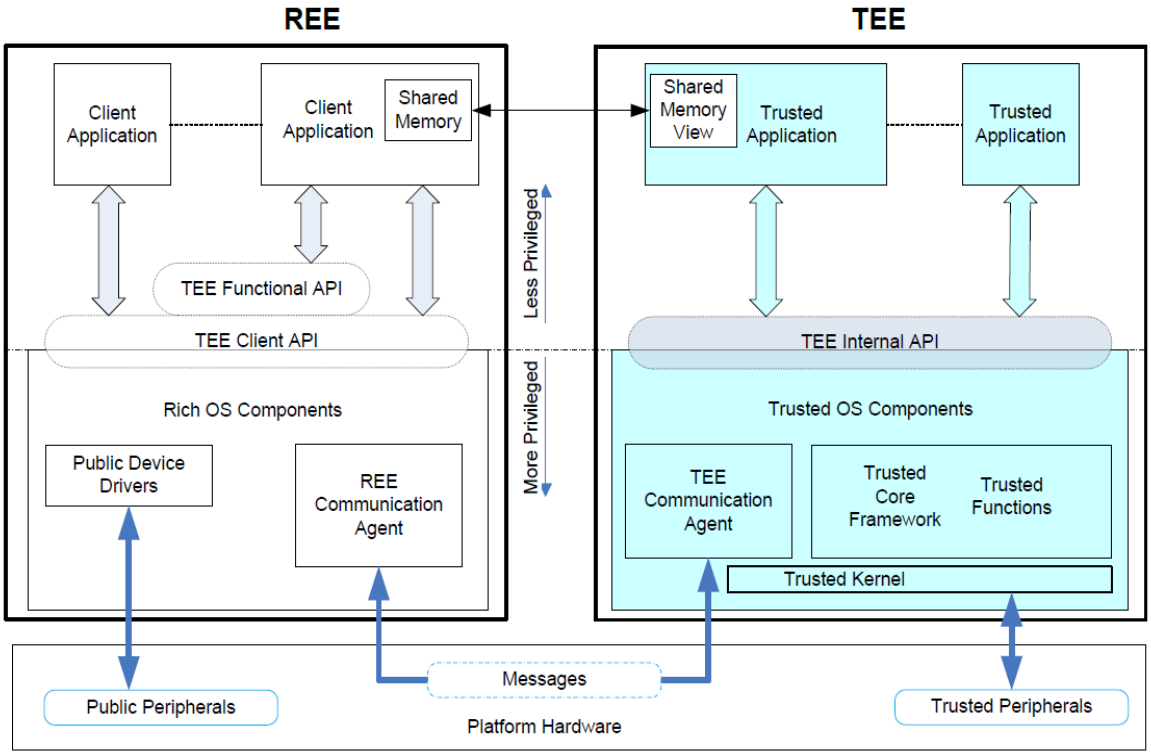

Figure 2: DRM Architecture [8]

Figure 2 shows a typical TEE architecture proposed by GlobalPlatform [8] - a cross industry, non-profit organization, which is publishing specifications for embedded system security. As shown in the figure, the TEE environment sits along with the Rich Execution Environment (REE) that hosts a normal OS such as Windows, Linux, and Android etc. The goal here is to enable Trusted Applications (which reside in the TEE) to provide isolated and trustworthy

capabilities for service providers, which can then be used through intermediary Client Applications (which reside in the REE).

To support a trusted environment, the REE contains three components: the TEE Functional API, the TEE Client API, and the REE Communication Agent. The TEE Functional API offers a set of OS-friendly APIs to the Client Applications. The offered APIs will be used to access some TEE services, such as cryptography or trusted storage. The TEE Client APIs are low level communication interfaces, which are used by the Client Applications to exchange data with the Trusted Applications. The communication tasks are handled by the REE Communication Agent and the TEE Communication Agent over a secure channel handled by the TEE.

Within the TEE, two classes of software: the hosting code provided by the Trusted OS Components and the Trusted Applications, create the trusted environment. Besides the TEE Communication Agent, the trusted OS components consists of the Trusted Core Framework (which provides OS like functionalists to the Trusted Applications) and the Trusted Functions (which provide the supporting facilities to application developers). The Trusted Applications, on the other hand, are the point of contact for the Client Applications. A Trusted Application creates a communication session for a Client Application. The session has its own state containing the session's context and the contexts of tasks executing within the session. The Trusted Application owns a separate physical memory space than physical memory spaces of other Trusted Application. Any large amount of data between the Trusted Application and the Client Application is exchanged by using a memory accessible to both Trusted Application and Client Application, such as a shared memory.

2.2 Current DRM schemes

Industry and academia are working on DRM protection schemes for quite a while. To this end, both hardware-based TEE solutions and hypervisor virtualization-based isolation techniques have been proposed. Most popular CPU architecture vendors, such as ARM, Intel and AMD, have provided hardware support for both TEE and virtualization.

In this section, we will focus our discussion on the available TEE solutions provided by Intel and ARM in Section 2.2.1 and Section2.2.2 respectively. For completeness, we will further discuss about two more old TEE architectures: the Texas Instrument's M-Shield and academia's AEGIS in Section2.2.3 and Section 2.2.4 respectively.

2.2.1 Intel-based systems

2.2.1.1 Intel TXT

Intel Trusted Execution Technology (TXT) is an old hardware-based Intel TEE scheme based on a TPM chip [11]. The main objective of Intel TXT is to provide users a secure way of loading and executing system software in a potentially compromised machine. This goal is achieved by using the cryptographic functions of the TPM to confirm that a code will not be executed until it has been measured and verified to be correct.

The most general and widespread use of Intel TXT have been to protect and permit a verifiable secure installation, and to launch and use a critical software, such as a hypervisor or an OS, in a trusted environment. Although TXT could have been a suitable technology for DRM and content protection, in reality, this technology has not been used a lot for this purpose. Some of

the reasons for this non-use are: (i) TXT can only provide launch-time protection (no run-time protection is offered); (ii) TXT is strongly dependent on an external TPM chip, which is not often shipped with general purpose computing devices, such as PCs; and (iii) as proved by Wojtczuk and Rutkowska [12], the security of TXT can be bypassed by infecting the System Management Mode (SMM) handler, which is the most privileged processor mode.

2.2.1.2 Intel SGX

The forthcoming Intel technology, Intel SGX, can be seen as the next generation Intel TXT [18]. This new technology is not dependent on a TPM to do measurements, sealing and attestation; but, the design contains a built-in extension that can emulate the TPM functionality. Unlike the Intel TXT, SGX can also provide data confidentiality and data integrity protection from privileged malware.

SGX provides a new mode of execution on the CPU, a new memory semantic, and a set of new processor instructions. Using the offered functionalities of this technology, a user can create an *enclave*, which is an isolated protected container similar to the ARM TrustZone's secure world. Unlike the case of TrustZone, one can, however, create multiple enclaves. Attempted access to an enclave memory from a non-resident application is prevented even for the privileged software such as hypervisors, BIOS, or the operating system kernel. Thus, SGX can be used as a DRM protection scheme.

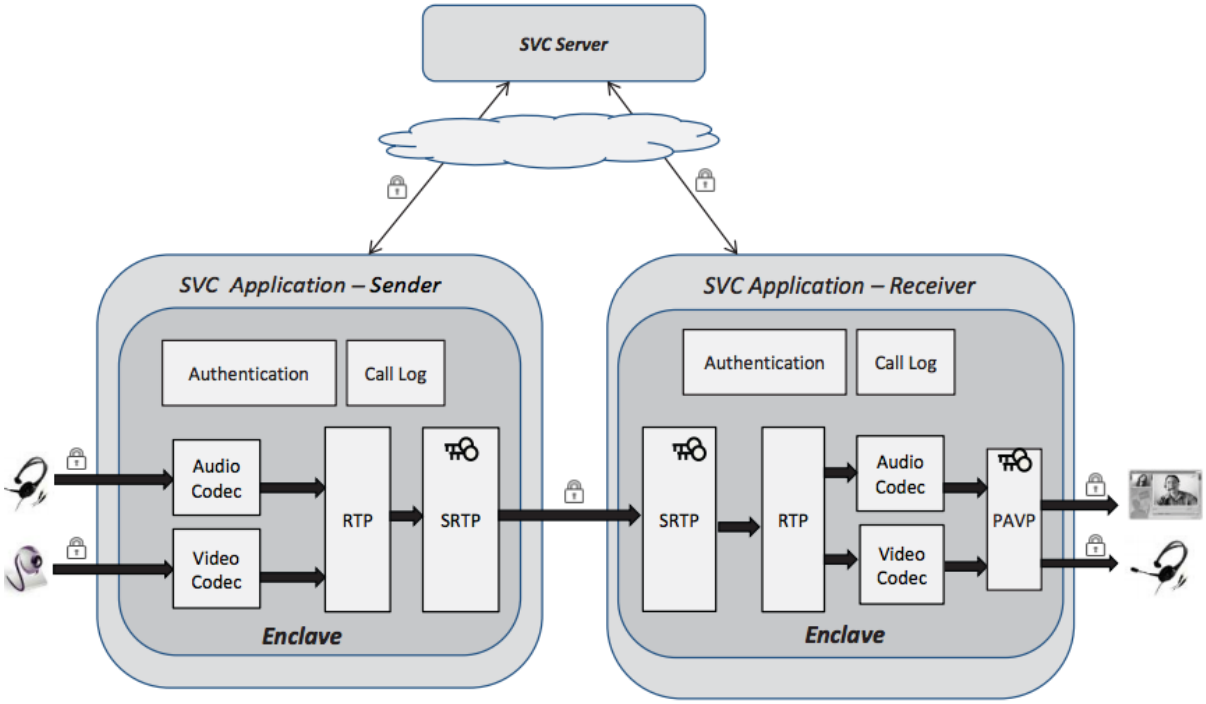

Figure 3: Secure Video Chat Application structure [18]

Recently, Intel researchers have demonstrated the DRM protection capability of Intel SGX by designing a prototype Secure Video Conferencing (SVC) application [19], which protects the video path by using SGX and Intel's Protected Audio Video Path (PAVP) (Figure 3).

The prototype SVC was built by modifying an open source video conferencing application to run in an Intel SGX enclave so that video conferencing operations can be isolated from rest of

the system. As shown in Figure 3, in the SVC application, the users first establish a call session using a call initiation protocol. Thereafter, the locally captured AV (Audio/Video) streams are transmitted using a transport protocol such as Real Time Protocol (RTP). The AV streams are encrypted for secure transmission using a secure transmission protocol such as Secure Real time Transport Protocol (SRTP). Since the path between the application memory and the video frame buffers are insecure, Intel PAVP technology is used to encrypt the audio and video streams. The next section provide an overview of Intel PAVP.

2.2.1.3 Intel PAVP

PAVP is a hardware feature integrated into the Graphics Media Accelerator of the newer Intel chipset, such as the Intel 4 Series Chipset Family and the Mobile Intel 4 Series Express Chipset Family, to ensure a robust and secure content protection path for high-definition video playback (e.g., Blu-ray discs) on newer Microsoft OS (since Windows Vista). This new Intel feature enables hardware accelerated decoding of the encrypted video stream to deliver smooth playback. One can enable or disable PAVP in the BIOS using three modes: (i) Disabled, (ii) PAVP Lite, which reserves buffer memory for encryption of compressed video data, and (iii) Paranoid PAVP, which reserves memory during booting (reserved memory cannot be seen by the OS).

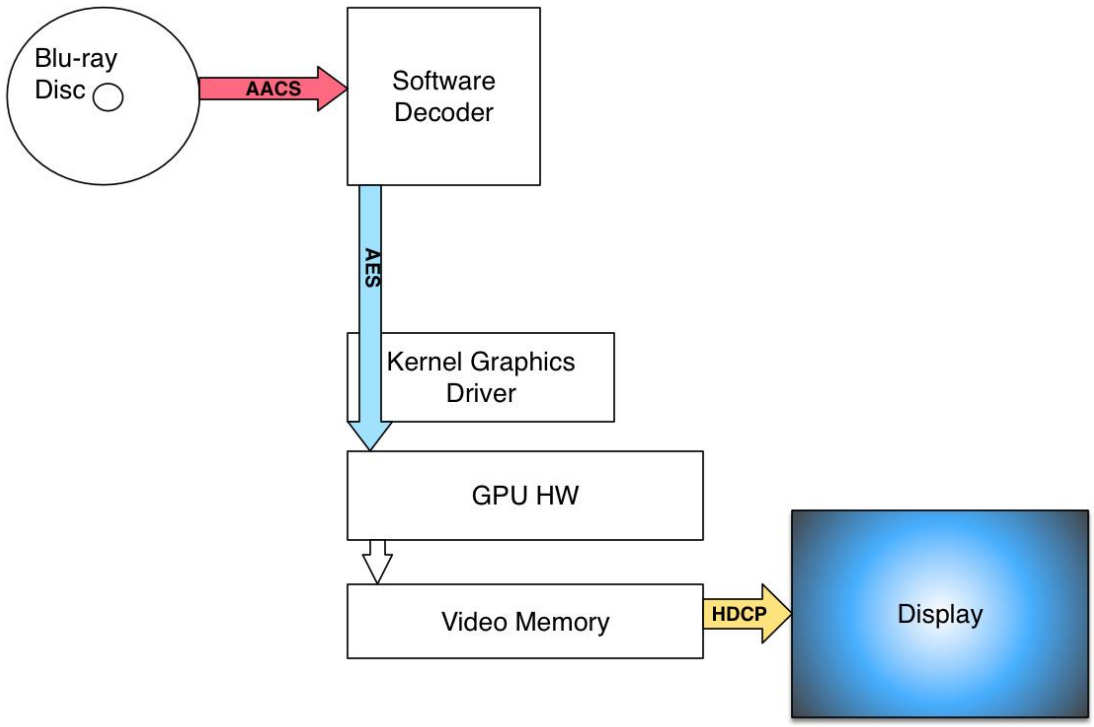

Figure 4: Protected audio video path

Figure 4 shows how PAVP can be used to protect media content in Blu-ray playback. In the Blu-ray playback, the media content is encrypted by the Advanced Access Content System (AACS) before being sent to the Software Decoder. With the absence of PAVP, the decrypted output of the Software Decoder is typically sent to the the video driver for display. Thus, one, by accessing the communication channel from the Software Decoder to the Display, can know the decrypted media content. When PAVP is enabled, such data leak, however, can be avoided

by re-encrypting the decrypted data (output of Software Decoder) with a symmetrical key encryption, such as AES-CTR (Advanced Encryption Standard Counter Mode), before passing it to the hardware accelerator of the GPU. The GPU then in turn decrypts and decodes the stream in hardware, and stores the decrypted video in video memory. In order to protect against snooping of uncompressed frames between the physical connector and the display device, link protection is used, which uses GPU HDCP to encrypt the entire frame buffer before sending it over the video output.

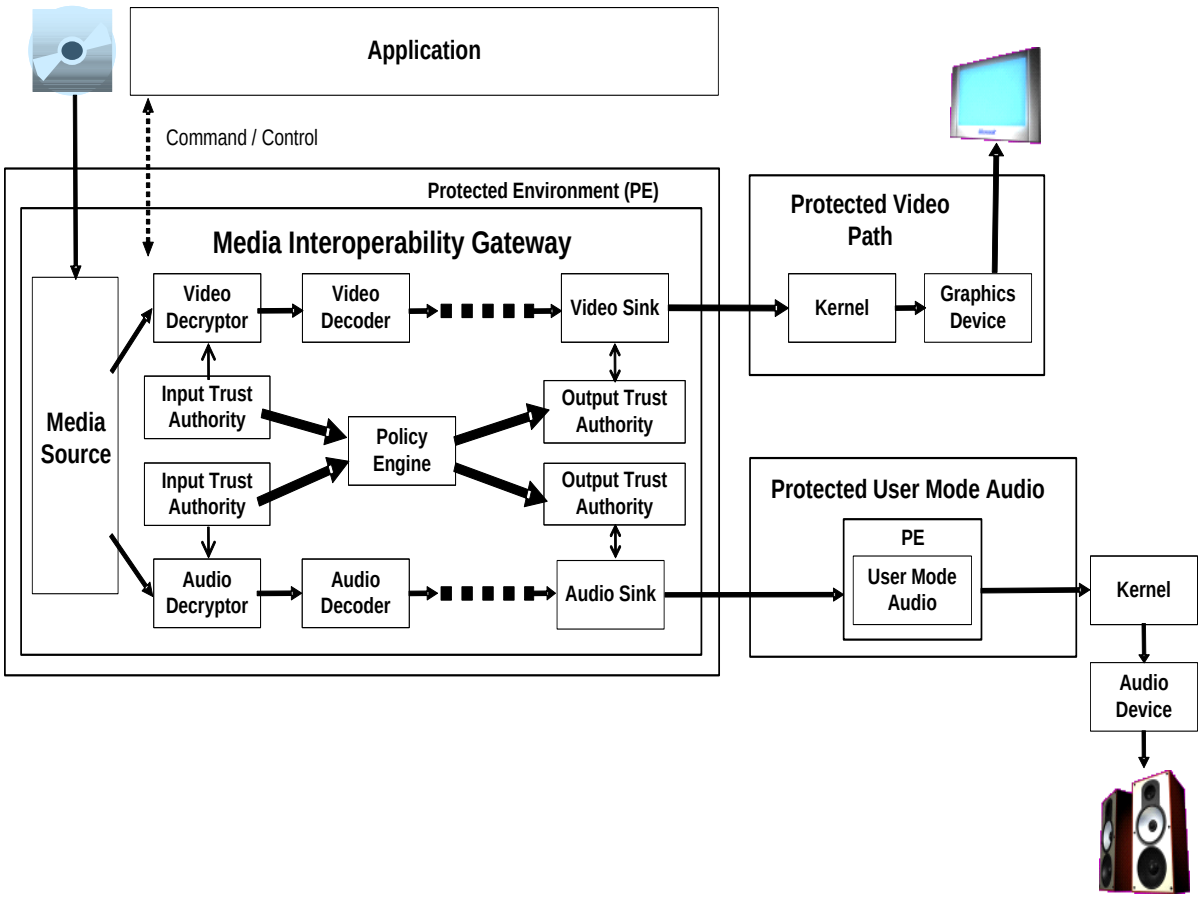

Figure 5: Window Vista's protected media path [20]

Figure 5 shows how Windows Vista uses for the Protected Environment (PE) [19] to provide a protected media path (called as Output Content Protection in Microsoft). The PE is the synonym of the TEE in Microsoft. As shown in the figure, the PE contains the Input Trust Authority, Policy Engine, and the Output Trust Authority, which enforce the policy on the protected content. The Input Trust Authority checks the user's rights to play the protected content, and provides the Video Decryptor the decryption keys. The Output Trust Authority enforces DRM policies by ensuring that all the security requirements are fulfilled before playing the file. The Output Trust Authority requires that digital outputs such as Digital Visual Interface (DVI) and High-Definition Multimedia Interface (HDMI) are protected with HDCP (discussed in Section 2.2.1.4.1). The Hardware Functionality Scan (HFS) is used to verify the authenticity of a device, and the PAVP certification is used to determine the correctness of the device driver.

2.2.1.4 Intel Insider

As part of the Intel HD Graphics, a new technology called Intel Insider is developed for protecting online HD movie content. Similar to Intel PAVP, this technology is a hardware feature in the GPU chip, designed to securely purchase and rent premium 1080p HD and 3D movies on the PC [15]. This technology gives the user the possibility to access movies on their PC in full high-definition from supported content providers. In order to provide end-to-end solution, Intel uses the DLNA CVP (Digital Living Network Alliance Commercial Video Profile) scheme, which is a standard for commercial content interoperability between media devices and DTCP-IP (Digital Transmission Content Protection over IP) for content encryption and the DRM. This scheme protects the output of the visual interface using HDCP [16]. In the following, we describe the HDCP in more detail.

2.2.1.4.1 HDCP

HDCP is used to protect digital audio and video data in transmission, such as HDMI, DP (Display Port), and DVI data. This scheme protects the content against eavesdropping and man in the middle attacks.

The HDCP protection scheme consists of three main elements [26].

- Authentication protocol
- Encryption
- Key revocation

First, the HDCP transmitter verifies that the HDCP receiver is licensed to acquire HDCP content. The Authentication and Key Exchange (AKE) accomplish this requirement, as the HDCP receiver's public key certificate is verified by the transmitter and the master key is exchanged. The protocol then enforces locality on the content by enforcing that the round trip time (RTT) between a pair of messages is not more than a specified duration. The HDCP receiver is then authorized to receive HDCP content, and the transmitter initiates a Session Key Exchange (SKE) for being ready to start data transfer. The transferred data will be encrypted with the Session Key.

After the legitimacy of the HDCP receiver being determined through the authentication protocol, the transmission of encrypted HDCP content can be started. The encryption method uses a 128 bit AES stream cipher. The transmitted stream of encrypted data is sent to the receiver, who can decrypt the content with a copy of the session key.

All transmitters or receivers include a Key Selection Vector (KSV), which is a 20-bit binary value that uniquely identifies the device. If the legitimate devices is compromised to permit unauthorized use of HDCP content, then the key revocation system in the HDCP system enables the HDCP transmitter to identify the compromised devices leading to prevention of the transmission of HDCP content.

HDCP is widely adopted in home theatre systems. The newest version 2.2, released in September 2013, is considered to be safest among all HDCP schemes. HDCP security, however, has been breached several times in the past. In 2001, cryptanalysts showed some fundamental security weaknesses of the HDCP technology. In 2010, a master key was leaked that can neutralize the key revocation feature of HDCP [44] by creating new ones when the old ones were revoked. This means that that unlicensed devices could simply use this leaked master key to dynamically create new ones, making revocation impossible. This hack was confirmed by

Intel, which threatened legal action against anyone producing hardware to circumvent the HDCP protection.

2.2.2 ARM-based systems

2.2.2.1 ARM TrustZone

The ARM processor is one of the most dominant processor architectures in recent embedded systems. ARM's processors are also being widely used in mobile smartphones. From its ARMv6 architecture, ARM introduced a new set of hardware based security extensions called TrustZone [21], which provides hardware level isolation for processing of critical data. A TrustZone powered processor core is split into two virtual isolated cores: one considered being secure and the other being non-secure. ARM calls a core as a *World*.

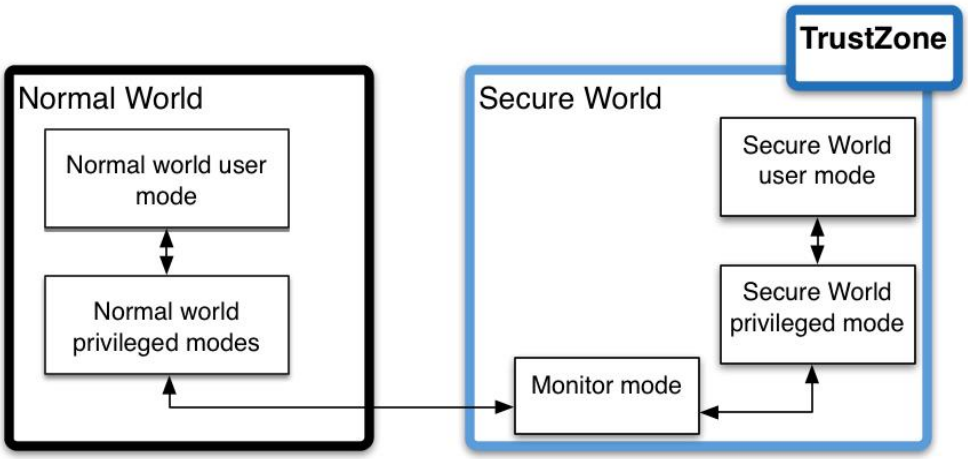

Figure 6: TrustZone split-world based isolation.

Since one world is isolated from another, one can put the most security critical applications in one world, and the general-purpose non-secure applications in another world (Figure 6). The CPU-state context is banked between the two worlds, and all the privileged and unprivileged modes found in the older ARM cores now co-exist in both worlds. In order to securely context switch between the two worlds, a new secure mode monitor mode has been introduced.

However, having a TrustZone enabled CPU core alone cannot provide a secure system, as all the communicating SoC (System on Chip) peripherals have to be TrustZone-aware to respect the security constrains imposed by the secure world. Designing a complete secure system requires careful SoC adaptations, and in the following section, we will go through the most significant customizations.

AXI system bus

In order to correctly utilize the TrustZone security extension, one extra bit, called as Non-Secure (NS) bit is added. This extra bit is propagated through all peripherals that can access the secure world. In other words, a peripheral must be TrustZone aware by the use of the extra bit. With the help of the NS bit, the TrustZone aware peripherals are now separated into two worlds, and any access request originating from to the normal world can be restricted to access the secure

world. This extended bus design is known as the AMBA AXI system bus, and can be found in TrustZone enabled SoC architectures [22].

Memory isolation
To isolate memory between the two worlds, the Memory Management Unit (MMU), address space controller (ASC), caches, and Tightly Coupled Memory (TCM) need to be TrustZone aware. The MMU provides each world with a local set of virtual to physical address translation page tables. The page tables in the secure world are protected to prevent any access to it from the normal world. The TrustZone ASC is used to partition its slave address range, such as DRAM (Dynamic Random Access Memory), into several distinct secure and normal regions. TCM, which is a dedicated memory area located in the same physical package as the CPU, is also designed to contain secure or non-secure data and instructions. TCM is however less common in today's smartphones, as on-chip memory is more prevalent in mid and high-range CPU's today.

DMA protection
Direct Memory Access (DMA) controllers also need to be TrustZone aware as certain peripherals use the DMA to transfer data. The DMA controller works directly with the physical memory addresses and thus bypasses any MMU translation or access permissions. In order to protect against DMA attacks, one would need a TrustZone aware DMA controller together with the TrustZone ASC to prevent a peripheral assigned to the normal world performing a DMA transfer to/from secure world memory regions.

Interrupt controllers
Hardware interrupts can both be handled in the normal world and the secure world. For TrustZone solution, secure world interrupts must be isolated from the non-secure world interrupts. The TrustZone-aware interrupt controller fulfills this requirement by handling secure interrupts in the secure world and non-secure interrupts in the normal world. A non-secure interrupt can never interrupt an application in the secure world.

TrustZone design
The ARM TrustZone hardware platform architecture is only a specification, leaving the SoC designers free to customize a specific implementation to suit their business needs. It could mean that the OEMs (Original Equipment Manufacturer) could leave out components that they consider too expensive to mass-produce, severely constraining some security features or in worst case open up for attacks. For example, excluding a TrustZone aware DMA controller would open up for DMA attacks that can transfer data from secure world memory into the normal world. However there is always a tradeoff of how much security you need and how much you are willing to pay for it. The OEM could carefully design the SoC to have system-wide security with every potential dangerous peripheral to be TrustZone aware and protected. However, the targeted market has to be willing to pay for the added expenses.

2.2.2.2 ARM hypervisor mode

The latest ARM cores, such as the Cortex A15 provides support for virtualization extensions using a new execution mode called HYP (Hypervisor) that is more privileged compared to the standard supervisor (privileged) and user (unprivileged) mode (Figure 7).

With the help of the new mode and the MMU, hypervisor software residing in the HYP mode can provide support for multiple guest OS running on the same hardware. As the hypervisor runs in a more privileged mode compared to the guest operating system in the CPU, it is also

possible to create flexible custom protected areas for sensitive code and data, which can be unreachable by the main rich OS.

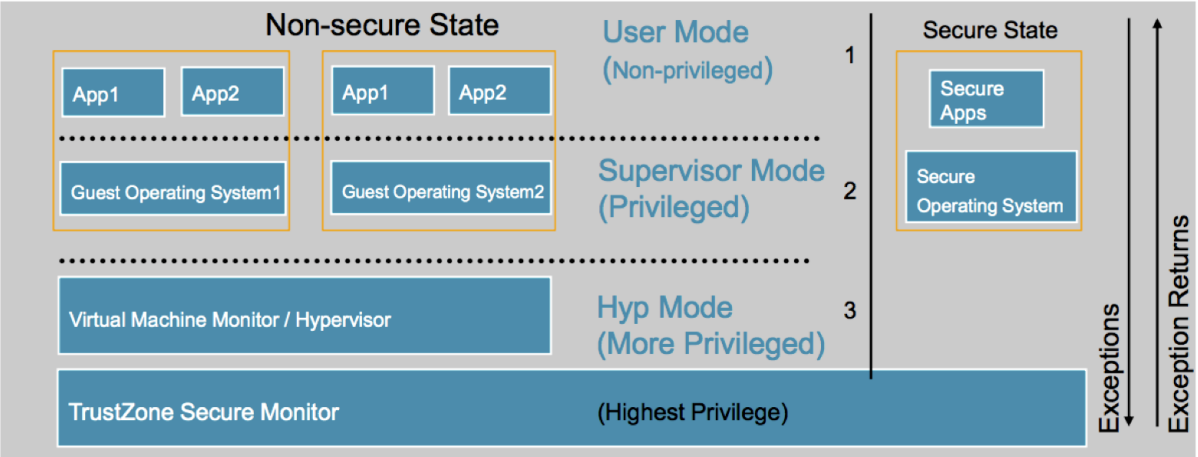

Figure 7: ARM Hypervisor extension support

Newer architectures contain new mechanisms to make virtualization easier with less performance overhead. This includes guest specific mechanisms for page table management, virtual interrupt handling, and making all CPU features virtualizable.

In contrast to TrustZone technology, the Hypervisor is not constrained to two worlds as it can create an arbitrary number of execution environments for its guests to run. Instead of a hardware NS bit, the hypervisor uses the CPU and MMU to distinguish between secure and non-secure requests. For example, the hypervisor uses the MMU to divide memory into secure and non-secure regions, and regulate memory access to secure regions. Such a solution, however, cannot prevent an unauthorized BUS master to access a protected memory region of a peripheral. An example is the DMA attack, which circumvents the CPU MMU protection. This issue, however, can be addressed by using a hardware device called System MMU (SMMU), which can provide protection functionalities to a DMA capable non-CPU peripheral, such as GPUs and Video Engines [41].

2.2.2.3 ARM Mali-V500

On June 2013, ARM announced the new Mali-V500 [9] hardware video decoder that has been architected with the support of the TrustZone technology to enable hardware-backed security for movie and TV content from their download to display.

In contrast to the general DRM schemes, which only protect the cryptographic keys, Mali-V500 can protect the content by providing a TrustZone-aware architecture that is specially designed to protect the media pipeline path. Thus, Mali-V500 can protect media starting from decryption to decode, and from decode to output to the external links. In order to take advantage of this solution, ARM has specified required SoC adaptions, which requires the TrustZone supported peripherals to be adapted to the target setting.

Figure 8 shows a high level design of a secure media path using Mali-V500, Mali Display and Composition module, and the TZC400 controller [10]. According to the TrustZone specification, the design contains a normal world and a secure world. The high level operating system (HLOS) Android is run in the normal world, and a secure OS is run in the secure world.

The secure world contains security critical applications and resources, such as DRM applications, cryptographic keys, and initialization modules of crypto hardware. Communication between these two worlds can only be carried out through a well-defined standardized Global Platform API, which ensures portability of the solution across different smartphones and platforms.

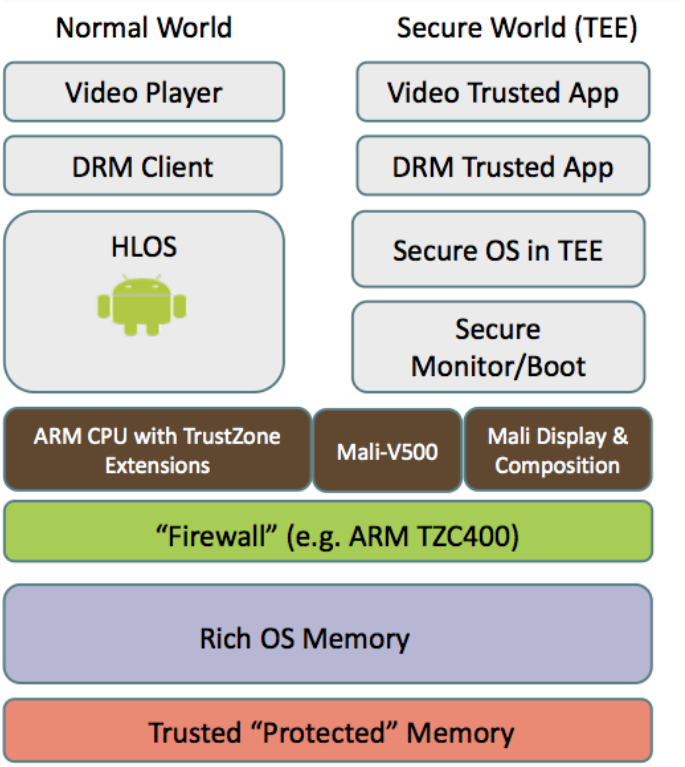

Figure 8: ARM Mali v500 TrustZone architecture [10]

The TrustZone address space controller 400 (TZC400) and Non-Secure Access IDentity input (NSAID) are used to partition the peripherals and memory into several isolated regions. For example, memory buffers used in processing, decoding, mixing and rendering of media can be protected by TZC400, and therefore can be made inaccessible to the HLOS. Thus any unauthorized access to secure memory regions can be avoided. As a result, secure world applications can only be made to utilize the Mali-V500 to decode and display copyrighted media content. By protecting output to the internal TrustZone aware display or to an external device via protected links such as HDCP and DTCP, the ARM specified Mali-V500 architecture can provide a secure video path.

Figure 9 shows how Mali-V500 can be used along with TZC400 and the NSAID to protect media. The blue boxes in the figure show peripherals aware of TrustZone, and the yellow boxes show peripherals protected by TZC400. TZC400 takes help of NSAID bits to partition the DRAM into isolated regions so that sensitive data cannot be accessed from the rich OS.

Another way of protecting the media content in the Mali-V500 architecture is by utilizing an ARM processor with virtualization extensions, such as the Cortex A15. Such processors possess a new HYP mode that can be used to protect software components and sensitive resources such as decrypted/decoded media content. Figure 10 describes the hardware architecture where the blue boxes represent TrustZone aware peripherals, and the yellow boxes represent hypervisor

protected peripherals and memory. Since a peripheral having DMA support is not protected by the hypervisor, SMMU is attached in front of each peripheral having DMA capability.

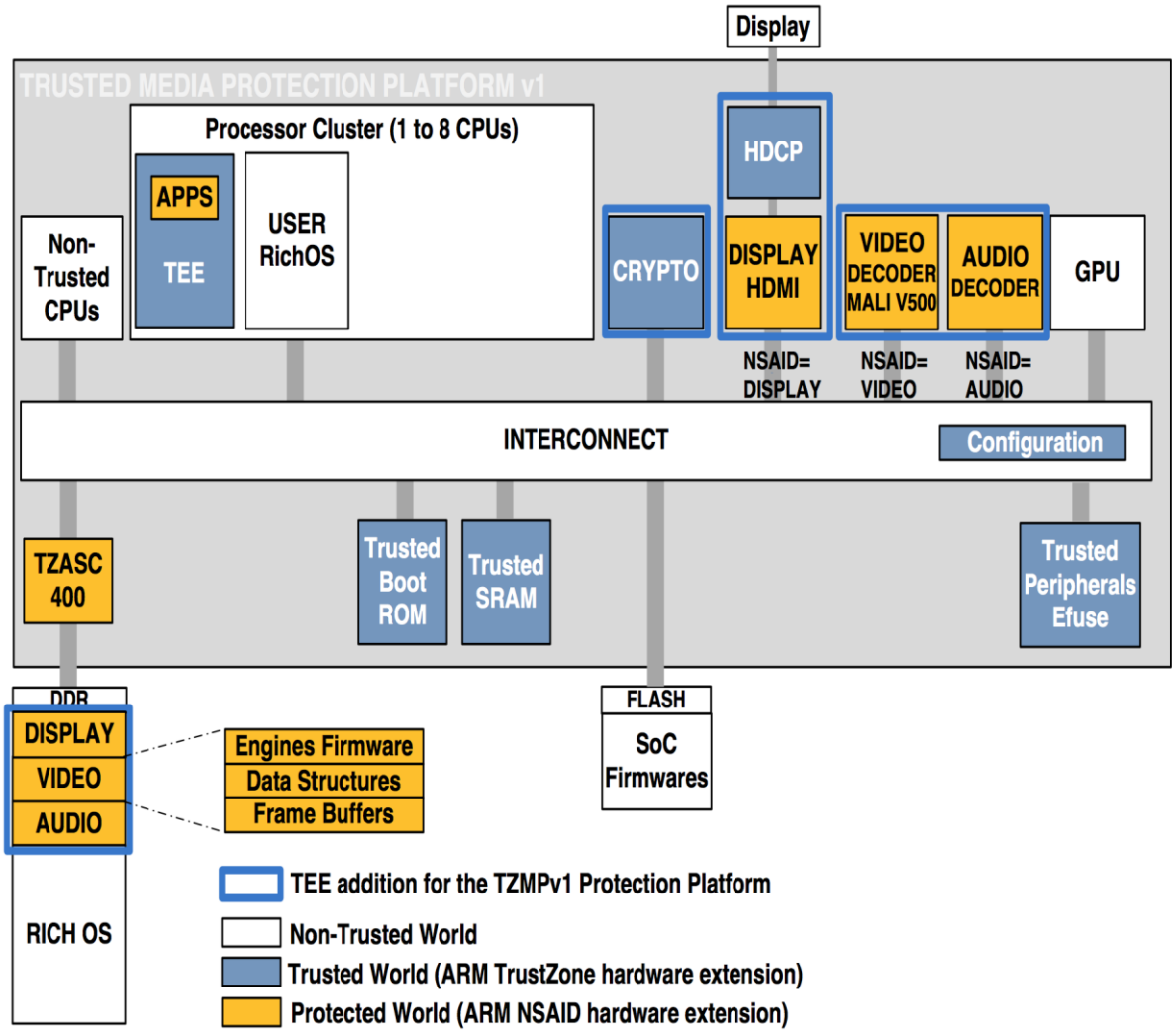

Figure 9: ARM Mali-V500 HW TEE & NSAID reference specification for Media protection [27]

To provide a full hardware solution for protecting the media pipeline, both the virtualization and non-virtualization based solutions require challenging adaptations to both SoC level and software level. Such a requirement is challenging to the smartphone industry since the solutions must be delivered in shorter time period.

Note that we did not find enough literature about Mali-V500 and its applications. Therefore, this document provides a cursory treatment to these technologies.

2.2.3 Texas Instrument M-Shield

In early 2000s, Texas Instrument cooperated with Nokia to design M-Shield [36] - a mobile security technology that can provide DRM functionalities. In the early version of M-Shield [38], program isolation is handled by the security monitor, which is a hardware logic external to ARM processing core but internal to the chip. The security monitor, which can only be triggered by executing a secure mode entry ROM (Read Only Memory) program, enabled

isolation by controlling some of the memory address bus lines in the ASIC (Application Specific Integrated Circuit).

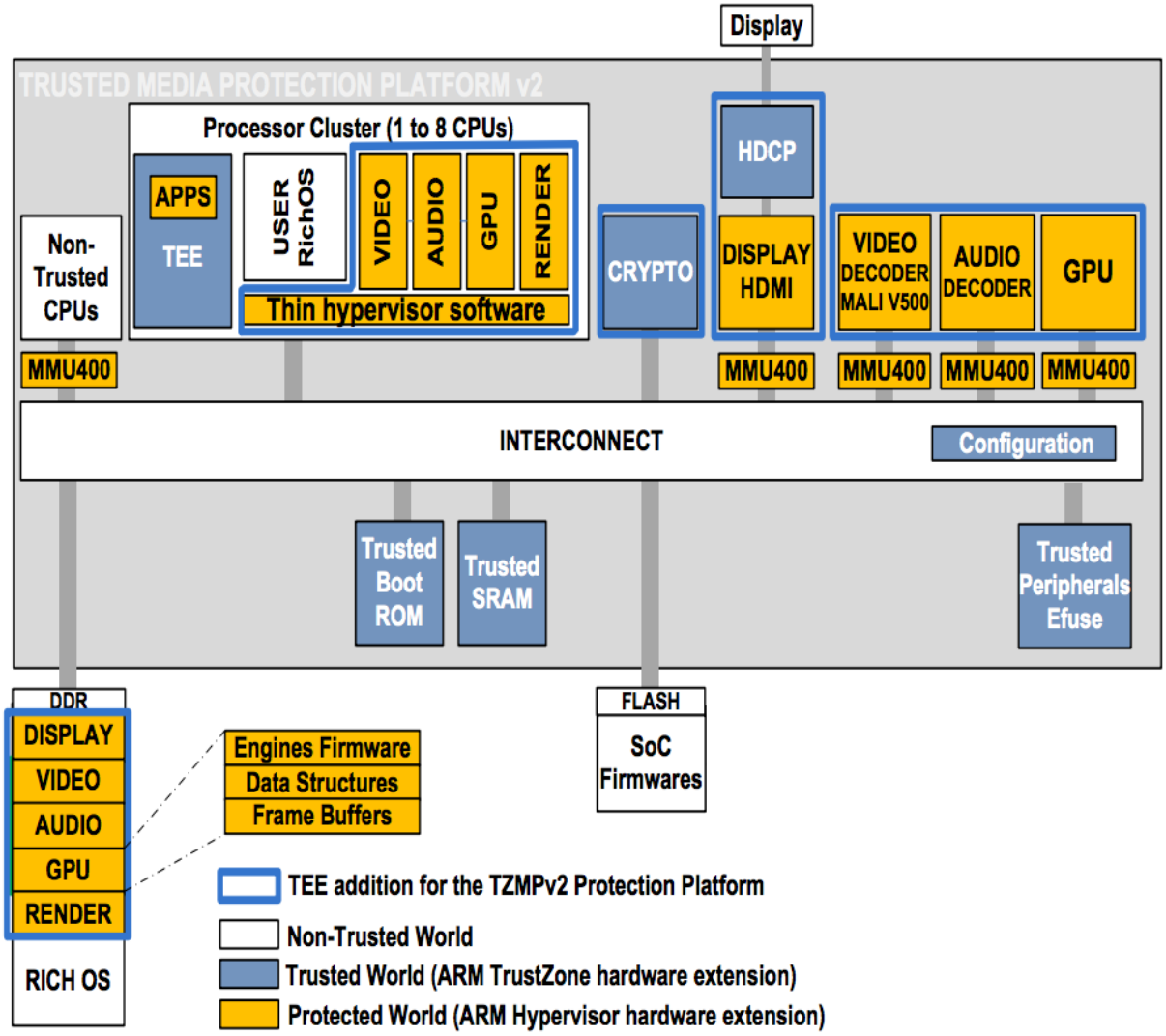

Figure 10: ARM Mali-V500 HW TEE & Hypervisor reference specification for Media protection [27]

Safe execution of sensitive applications and secure storage of important data is enabled by the secure environment. For example, addressing of on-chip memory and e-fuses is possible only when secure mode is on. The hardware part of the security monitor includes a security watchdog timer, a hardware random number generator (RNG), and a few other hardware primitives. The software part of the solution consists of the monitoring entry code that protected against intentional information leakage by flushing the virtual memory and disengaging the interrupts. Recent version of M-Shield [36] uses ARM's TrustZone. Figure 11 shows the high level working of recent M-Shield technology. M-Shield provides a secure environment by embedding a secure state machine (SSM) as well as the secure ROM and RAM. The SSM applies and guarantees the system's security policy rules while entering, executing, and exiting from the secure environment. DRM also provides a secure DMA to ensure the confidentiality of sensitive data (such as DRM-protected content) during their processing and transfer throughout the platform. A secure chip-interconnect allows peripherals and memories access

only by the secure environment and/or by secure DMA channels so that the confidentiality of sensitive information is guaranteed through the entire data path, from origin to destination. To validate and authenticate various software before execution, M-Shield uses public-key encryption, such as hardware-based AES, DES/3DES, SHA and MD5.

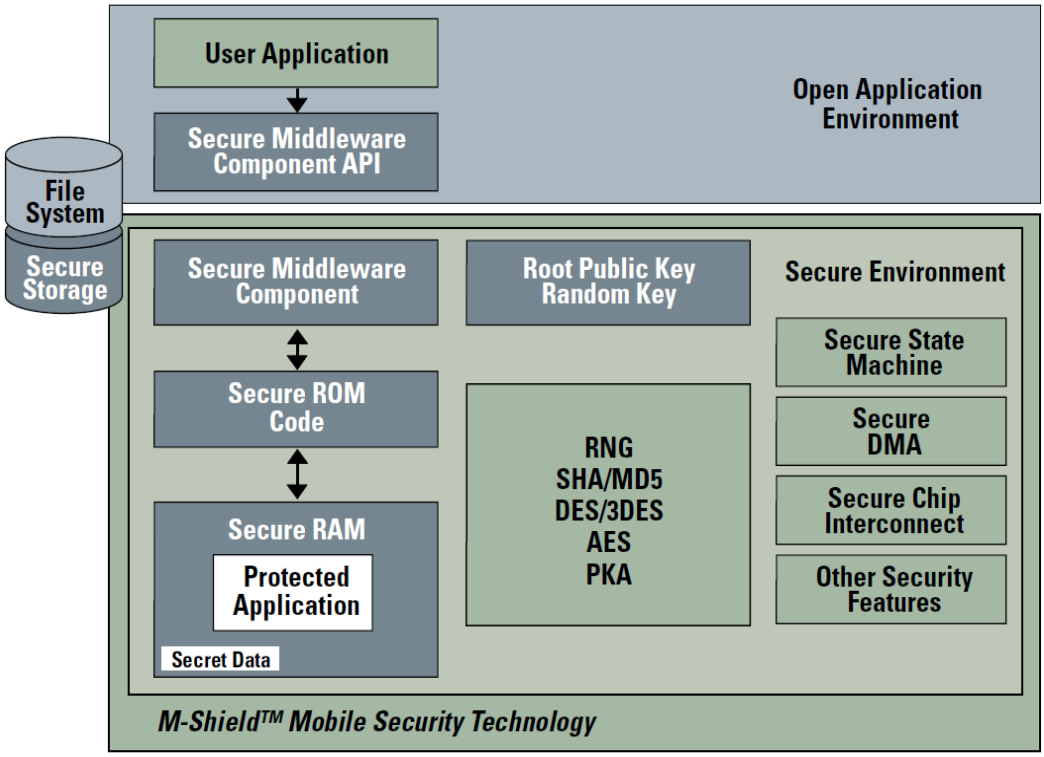

Figure 11: M-Shield Technology [36]

M-Shield's system level security is complemented by a security middleware component that includes a security framework and TrustZone standard-based APIs. This security middleware component enables interoperability, faster development and streamlining of security-based applications.

2.2.4 AEGIS

AEGIS [39] is an architectural design of a single chip secure processor proposed by the academia in early 2000. This design can counter both software and hardware attacks, and provides either a *tamper-evident* (TE) or *private and authenticated tamper-resistant* (PTR) environment. TE provides an authenticated environment by detecting any physical or software tampering, and PTE provides a secure and privacy aware environment by additionally denying information about software or protected data to the adversary. PTE has also been used to protect the DRM data.

- *enter_aegis*: start execution in TE/PTR environment
- *exit_aegis*: exit TE/PTR environment and return to normal processing
- *sign_msg*: generate a signature of message and a program hash with the processor's secret key

- *set_aegis_mode*: enable or disable the PTR environment. Set the static key K static that is used to decrypt static content corresponding to instructions and data that are encrypted in the program binary.

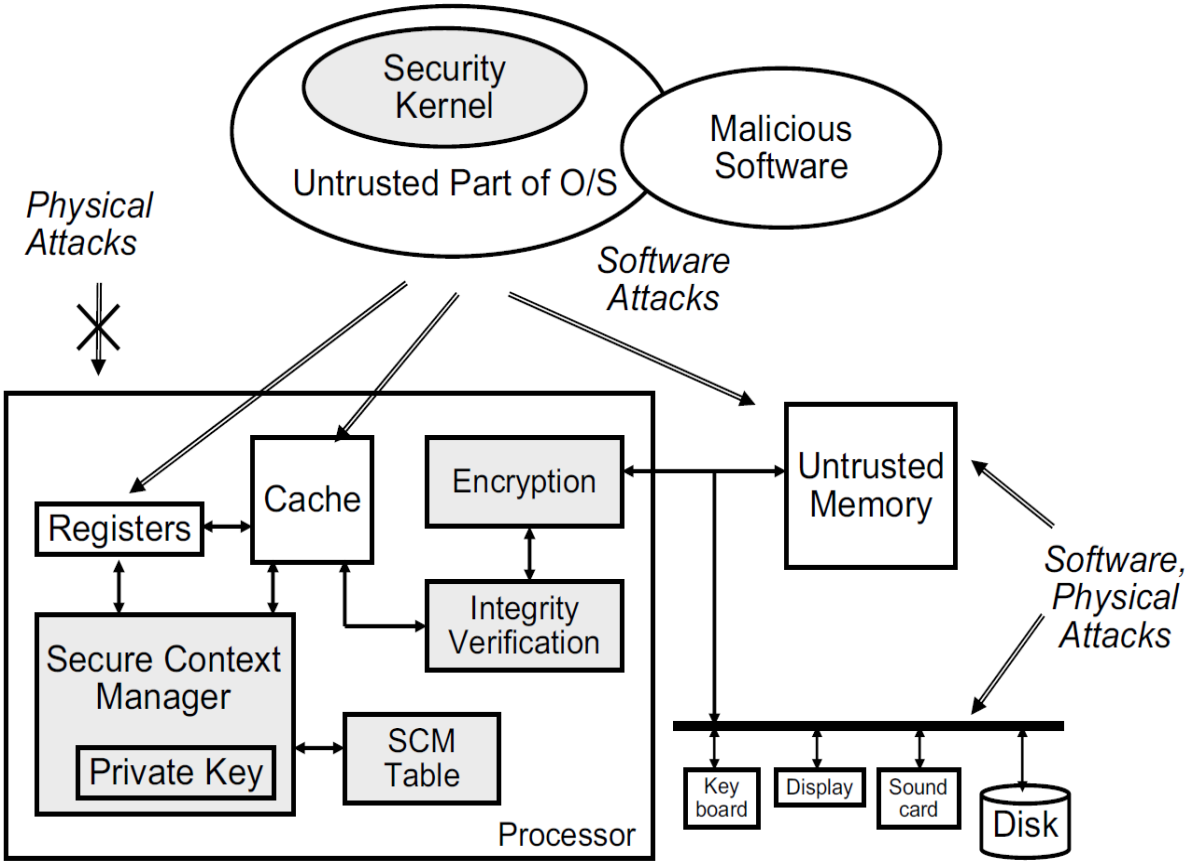

Figure 12: AEGIS Architecture [39]

Figure 12 shows the architecture of AEGIS that has been designed with the assumption that the processor is trusted and protected from physical attacks, so that its internal state cannot be tampered or observed directly by physical means. The external memory, peripherals, and the operating system, however, are untrusted.

The processor can securely communicate with outside world by using secret information such as Physical Random Function, or the secret part of a public key pair protected by a tamper-sensing environment. The processor also checks the integrity of the off-chip memory since the off-chip memory can be tampered with. In the case of PTR environments, the data is encrypted (to protect its confidentiality) before being stored in the off-chip memory.

AEGIS has been implemented by two different ways: with a *security kernel* (part of OS assumed to be trusted) and without a security kernel. The security kernel-based implementation provides more flexibility and requires less architectural modification, and the security kernel-less implementation provides better performance by putting all the mechanism in the processor and reducing the verification of trusted code. In the security kernel-based solution, the security kernel operates in a higher protection level than other part of the operating system, and therefore forms a Trusted Computing Base (TCB) along with the secure processor. Once booted by the processor, the security kernel takes care (such as starting, executing, and exiting) of the

processes that need to run in the AEGIS mode. On the other hand, in the security kernel-less implementation, the processor uses a specialized component, called Secure Context Manager (SCM), to keep track of the processes that are running in the secure domain. The SCM maintains a table called SCM table that holds various protection information for each secure process running in AEGIS mode.

AEGIS provides the following operations for an application program. The first three operations are required for both TE and PTR, while the last one is only required for PTR.

An application program enters the AEGIS (i.e., secure) mode by executing the enter_aegis instruction. The instruction specifies a stub region, which is used to generate a program hash (H(Prog)) that identifies the program. This program hash is stored in protected storage for later use. The stub code is responsible to check the correctness of data, any other code, and the environment, on which the application program will run. After stub completes this initiation process, the application program starts execution. During the execution, the TCB protects programs states in on-chip and off-chip memory by using integrity verification. The TCB also securely handles interrupts by protecting and preserving the register state of the program.

The sign_msg instruction is used to transmit a secret information from the system to the user over an untrusted communication channel. This operation returns the signature $\{H(Prog),M\}_{SKp}$ for a message M, where H(Prog) is the hash of the program that was computed when the AEGIS instruction was executed, and SKp is the secret part of a processor's private/public key pair. Therefore, when the user receives a message signed by the processor's secret key SKp, she knows that the message is from a particular program (program authentication) running on a particular processor (system authentication).

The set_aegis_mode enables the PTR environment and provides the static key encrypted with the processor's public key (E_{PKp} {H(Prog), K_{static} }), so the static key K_{static} can be decrypted only by a trusted processor for a particular program. The processor decrypts the value and sets K_{static} only when the program hash matches the hash of the executing program. This K_{static} is later used to encrypt data for ensuring privacy.

PTR environment ensures privacy by protecting the values of registers and on-chip memories from software attacks, and by protecting the values of off-chip memories from both software and hardware attacks. For the later purpose, fast encryption and decryption is used using Kstatic and $K_{dynamic}$ as keys. The static key K_{static} is used to decrypt instructions and data from the program binary, and obtained from the set_aegis_mode instruction. The dynamic key $K_{dynamic}$ is used to encrypt and decrypt data that is generated during the program's execution, and randomly chosen by the TCB when enter_ aegis is called.

Figure 13 shows how PTR can be used to provide DRM protection. As shown in the figure, in a pre-processing step, the content provider creates a trusted program player that will be run on the customer's side. This program is then embedded with the public key of the content provider, and this embedded information is called as *Player*. In step 1, the content provider calculates hash of the *Player, H(Player)*, and in step 2, it sends H(*Player*) to the customer. In Step 3, the player is run at the customer end by using enter_aegis and set_aegis_mode instructions (i.e., in the PTR mode). After this step, the customer knows the public key of the content provider. In step 4, the customer uses a standard protocol such as Secure Socket Layer (SSL) and the sign_msg instruction to establish a bidirectional private and authenticated channel with the

content provider. Once this secure communication link is established, the content provider sends the content in Step 5. Finally, the customer decodes and plays the content in Step 6.

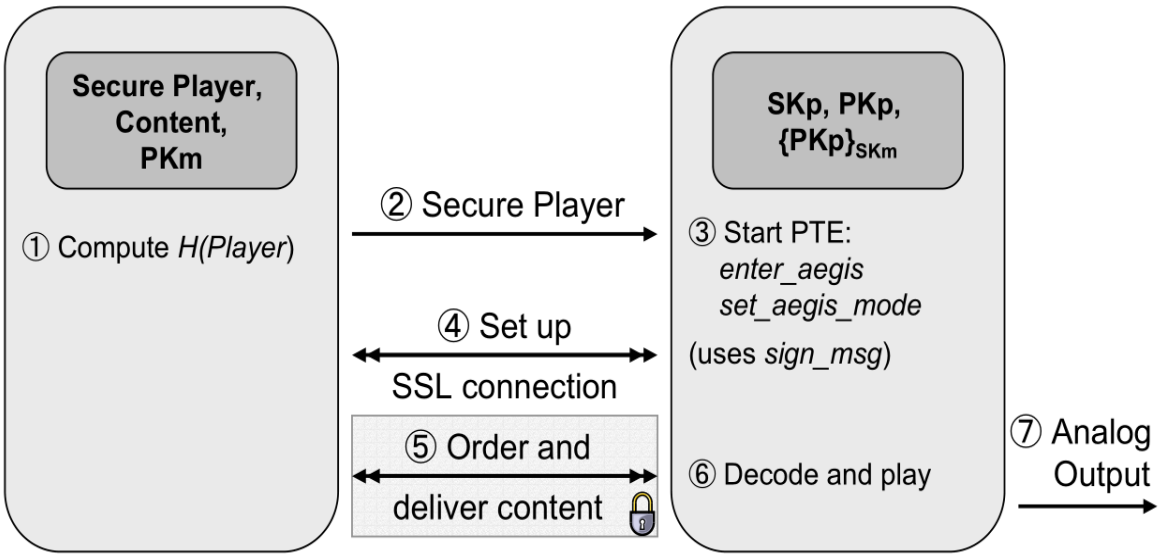

Figure 13: DRM using AEGIS [39]

2.3 Using Virtualization for DRM schemes

In this section, we will discuss three virtualization-based DRM schemes.

2.3.1 Video content protection

A recent, US patent application by Passera et al. [42] proposed a hypervisor virtualization based DRM scheme that can protect the media pipe without using hardware support to isolate the storage and processing of premium video content from other normal video content. The proposed scheme protects both encrypted and decrypted video content on mobile devices from being accessed by an unauthorized user.

Both the decryption of premium encrypted video, and the transfer of decrypted video to video memory for being displayed later on, are performed at the hypervisor layer (i.e., Virtual Machine Manager (VMM)). The hypervisor prevents the VM OS or VM applications from reading decrypted video content from the video memory by: (i) regulating the memory accesses by the processor (i.e., via MMU), and (ii) regulating the memory accesses by the DMA. The OS page table and/or the VMM page table is modified to divert the unauthorized access request to the video memory to a bogus data. To prevent DMA accesses requests to the video memory, the hypervisor also monitors the DMA requests to the decrypted video content.

Figure 14 shows the block diagram of the mobile device that uses the proposed virtualization-based media pipe protection scheme. As usual, the device contains a circuitry, which is responsible for data processing, and the system memory, which stores data.

The system memory stores two types of data: a VM and a hypervisor. The VM mainly contains an OS, device drivers, a DRM front-end, and other application softwares, such as video players, video libraries etc. The OS contains the page table that converts virtual addresses to *real* addresses (address space in hypervisor level), and can issue DMA configuration commands

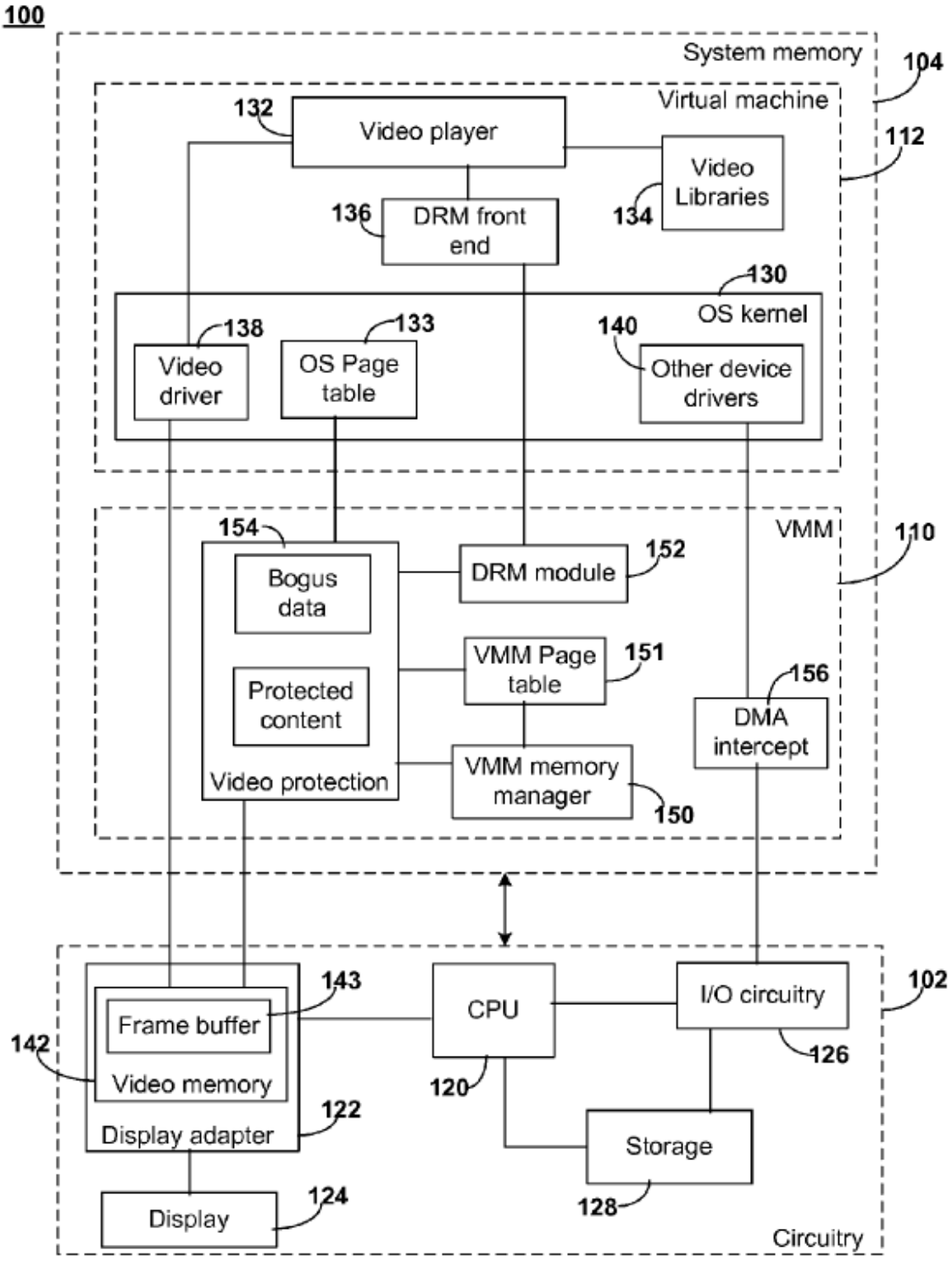

Figure 14: Block diagram of a mobile device using hypervisor-based media pipe protection scheme [42]

(which specifies the address of the device to be DMA accessed) via a device driver to a DMA-capable device. The hypervisor, on the other hand, includes a VMM page table, a VMM memory manager, a DMA intercept, a DRM module, and a video protection module. The VMM memory manager manages the access of a VM to the physical memory by taking help of the VMM, which converts real memory address to physical address. The DMA intercept module is

responsible to monitor the DMA requests from the VM to the video memory. The DRM module decrypts the encrypted premium video content received from the DRM front end. The decrypted video is then sent to the video protection module, which is responsible to store the decrypted video content in video memory. To disallow unauthorized access to the decrypted video, the video protection module updates the OS page table or the VMM page table to redirect the access to a bogus data (i.e., memory location other than the decrypted content). Among other components, the circuitry contains a display adapter that renders video in GPU, and stores the rendered data in the video memory. One can perform DMA access to the video memory by using the I/O circuitry component of the circuitry.

The program flow begins with the user requesting to play a video in the VM by launching the video player and selecting a video from the video library. If the selected video is not encrypted (i.e., not DRM protected), then the VM decodes the video content and stores the decoded content in the video memory. Otherwise, if the video content is encrypted, the DRM front end provides the video content to the DRM module of the hypervisor. The transmitted video content is stored in a storage in the hypervisor. The DRM module first checks the accessibility rights (i.e., the license) of the user; and if the user has rights, then the DRM module decrypts and decodes the protected video. The decrypted and decoded video content is then stored in the video memory by the video protection module. The video protection module modifies the OS page table and/or VMM page table so that the OS page table entries corresponding to the decrypted video content point to memory locations that contain the bogus data. The DRM intercept in the hypervisor intercepts all the DRM configuration commands, and discards the DRM requests to read the video memory. Therefore, access to the video memory (which stores premium content) by the OS or an application in the OS is prevented. Finally the decoded video content, which can be either premium content injected by the video protection module or non-premium content provided by VM, is rendered by the display adapter, and the rendered video is displayed by the video display module.

2.3.2 Managing stateful license

The license provided by the content provider to the user is often temporary in nature. For example, video content providers can restrict the number of times a user can access a video. The access rights can also include the type of allowed operations (such as copying, forwarding etc.) to the user. Such temporary license is called *stateful* license [3], since with times, these licensing information must be modified by the content provider.

For the storage and maintenance of stateful licenses, recently, Meng et al. [3] provided a dependable and scalable scheme that uses virtualization to isolate the license related operation from other DRM functioning operations. As shown in Figure 15, DRM licensing management procedures are put in a separate VM than the VM containing DRM procedures. Since license management is simpler, integrity of license can be guaranteed easily.

The workflow begins when a user wishes to render a DRM content (which is stored as a sealed digital resource) from the VM for digital resource access. Since a license to render the content is required, the license management procedure (which sits in the VM for manipulating license) is invoked. The license management procedure accesses user's sealed licenses, and accordingly, allows (or denies) the user to render the content. Thereafter, the VM for digital resource access unseals the content, and renders the unsealed content. At the same time, the VM for manipulating licenses updates the state of the licenses by invoking license management procedures. Note that all the license related operations, such as license import, update etc., are

allowed when the content provider is satisfied with the trust level of user's platform and the environment (such as hardware, VMM, license management procedures etc.).

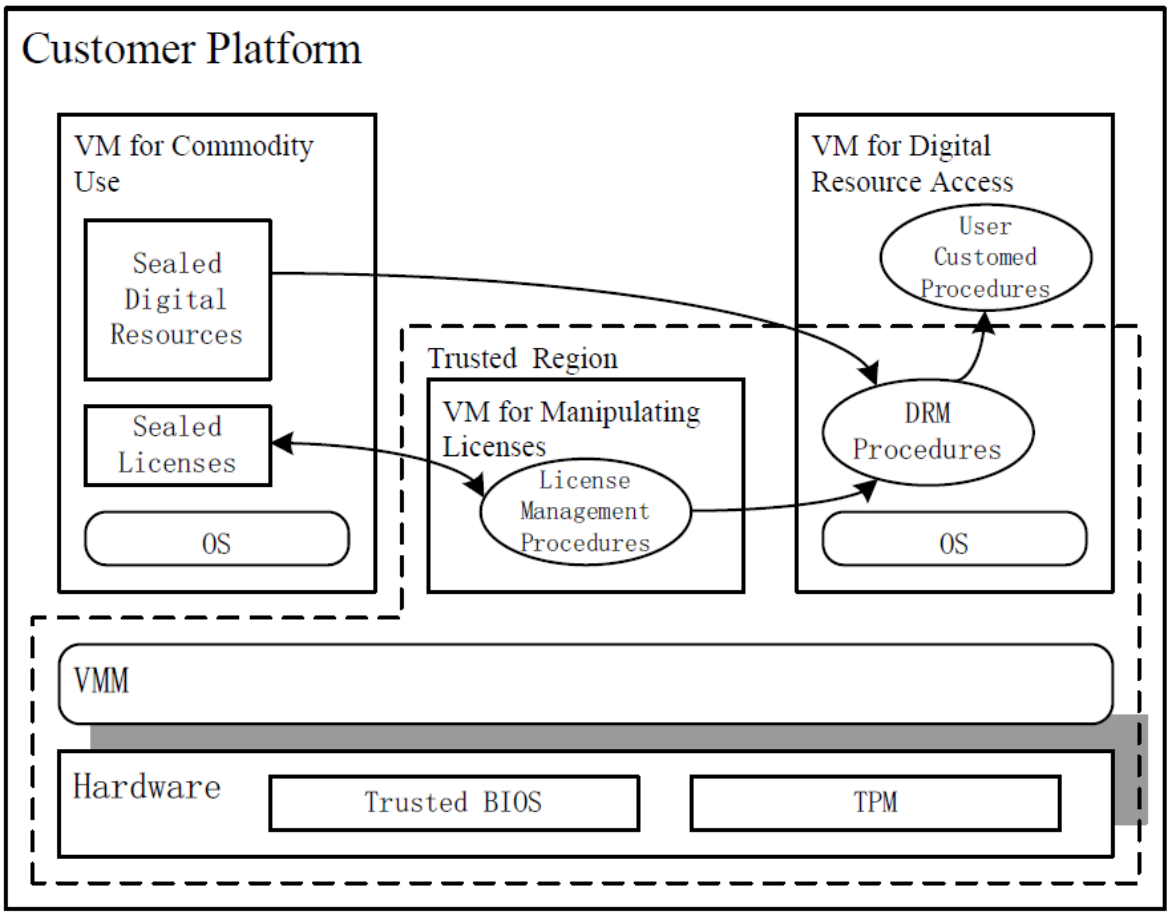

Figure 15: Isolation of DRM licensing operations [3]

2.3.3 vTPM

To provide VMs their own TPM environment (since number of VMs in a system are typically less than number of available TPMs), Berger et al. [28] proposed to virtualize the functionality of TPM. They first implemented the TPM functionalities, such as secure storage and remote attestation, in software, and then added functions to create and destroy instances of the virtual TPM (i.e., vTPM). The vTPM is placed into the hypervisor so that a VM can create an instance of vTPM for its own.

Figure 16 shows building blocks of vTPM. The vTPM consists of a vTPM manager and a number of vTPM instances. The vTPM manager is placed in a VM, which is secured by using the hardware TPM. Cryptographic capabilities, such as generating asymmetric keys, handling encryption and decryption etc., are handled by the vTPM manger. This vTPM manager creates vTPM instances and assigns a vTPM to a VM. Each vTPM instance implements all the functionalities of a hardware TPM. With the migration of a VM, associated vTPM is also migrated by the help of the vTPM manager (which encrypts the state of vTPM before migration).

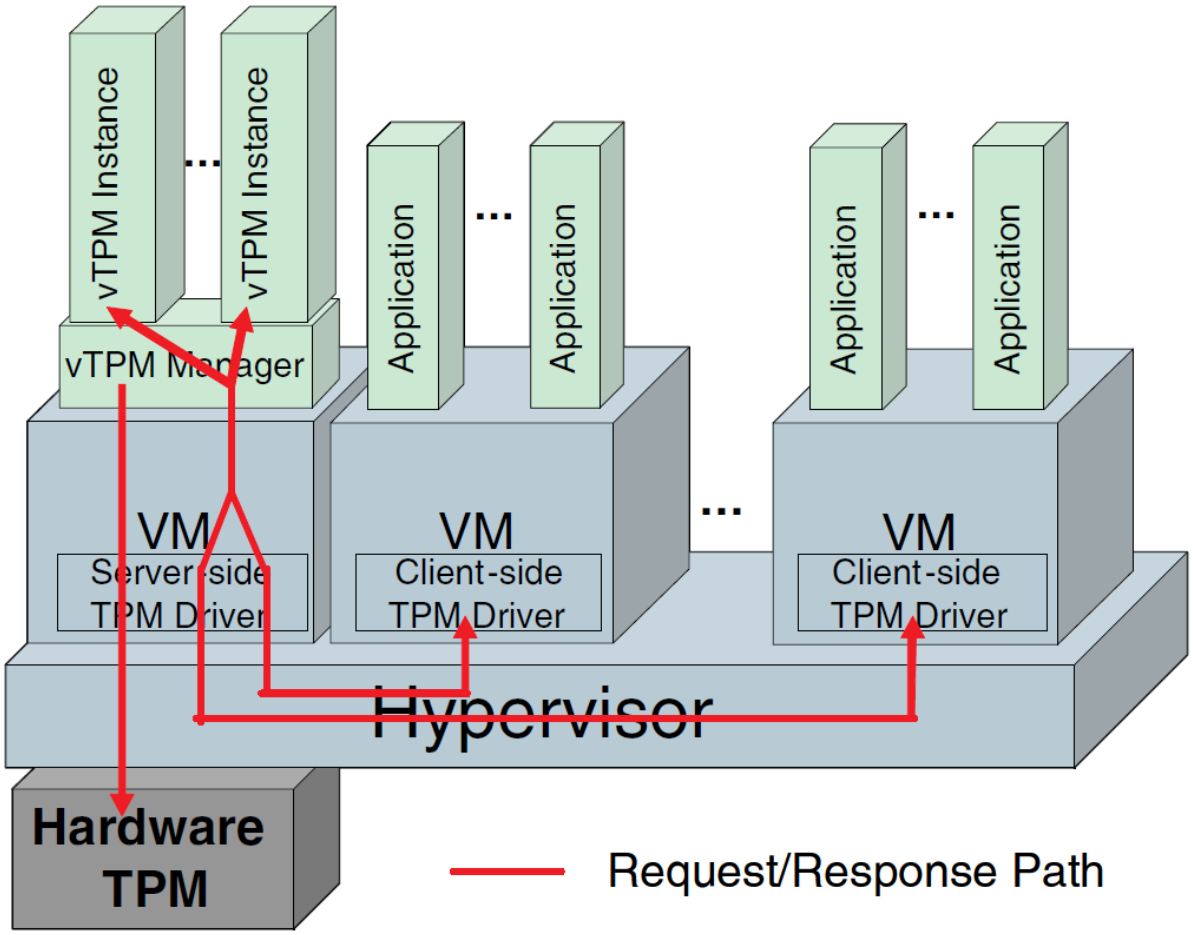

Figure 16: Architecture of vTPM

The VM communicates with the vTPM using a split driver model where the client-side driver runs inside the VM and the server-side driver runs inside the vTPM. The server-side driver helps in identifying a vTPM for a VM by prepending a vTPM identifier to each packet carrying a TPM command. The command's originating VM is determined from the unique interrupt number raised by each client-side driver.

3 Media Pipe Protection by Using Virtualization

In this section, we provide an overview of our ongoing work of using virtualization to protect media pipeline.

The core principle is to protect the media buffers that contain the clear text copyrighted media content with the help of the isolation properties that hypervisor software can provide. Figure 17 shows the tentative architecture of our approach. As shown in the figure, the DRM key handling and media content decryption can be executed inside the TEE TrustZone environment, and the media domain operations, such as media decoding, storing, can be performed with the help of the hypervisor (called as MEDH in the figure). To protect the clear-text media content from unauthorized bus accesses, such as unauthorized DMA accesses, we can protect a peripheral by putting the SMMU (System Memory Management Unit) in front of it. By going away with OEM hardware specific customizations, we can provide a more generalized solution. While we still need hardware support and modifications, this solution less intrusive compared to a fully

compliant TrustZone aware platform. Our approach is also advantageous for next generation smart phones that have hardware support for virtualization and SMMU protection.

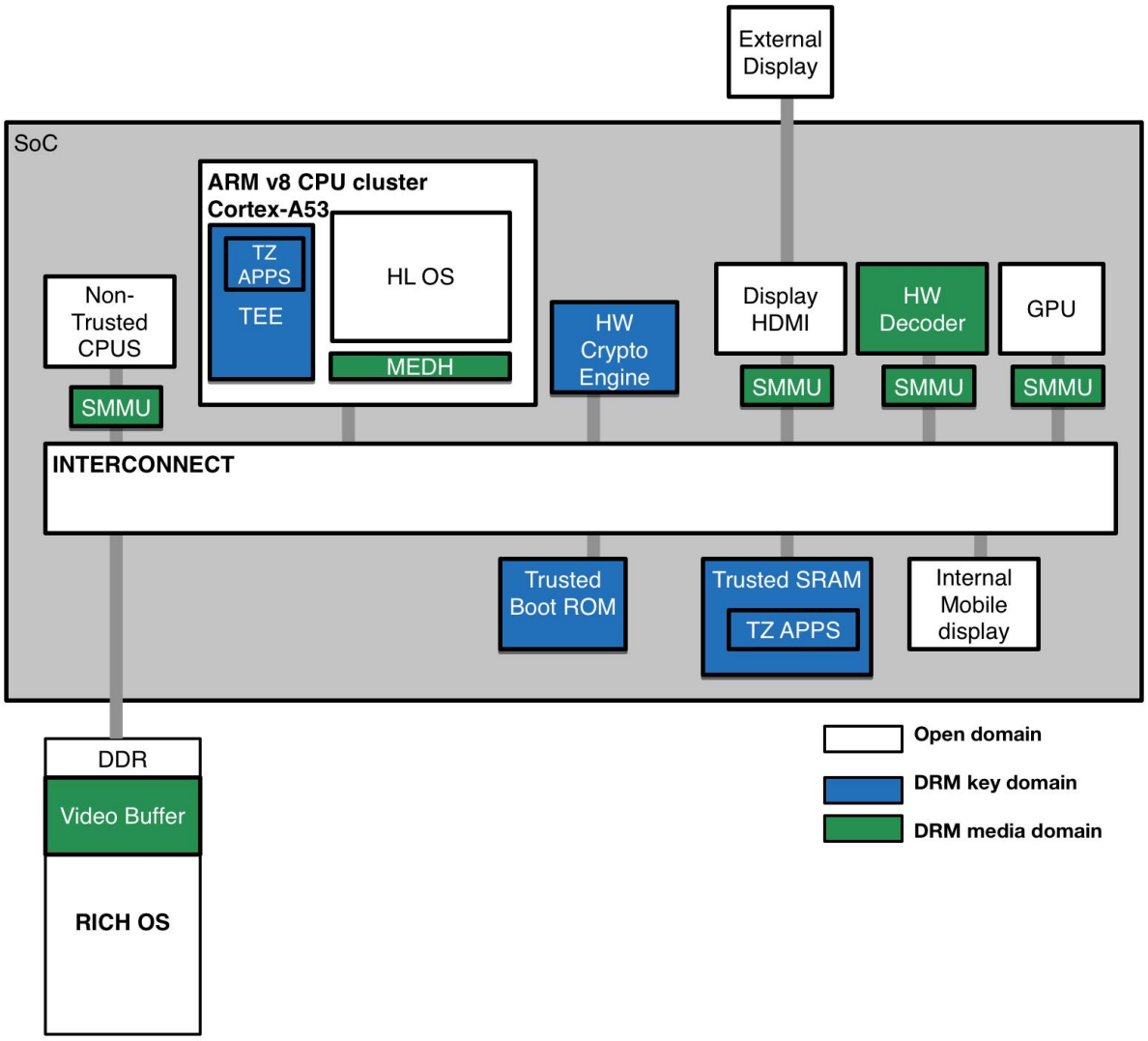

Figure 17: Proposed Architecture

In our approach, the hypervisor can utilize the new ARM architecture that comes with the hardware support for virtualisation. Thus, the guest OS can be executed with traditional operating system privileges, removing the need of employing modifications to the guest OS. With the help of the new page table extensions, the hypervisor can modify the translation of the clear text video buffers, effectively protecting the copyrighted video content from the general purpose OS. This technique is very similar to the current way of protecting the video buffers through the help of a secure memory handler, except that the proposed solution is less dependent on vendor customizations.

4 Conclusion

In this report, we studied the feasibility of using virtualization along with ARM TrustZone to protect the media rendering pipeline in a mobile device. To this end, we investigated how this problem has been addressed on different platforms and CPU architectures so far, and then discussed how virtualization technologies can be potentially used to protect the media pipe on mobile platforms.

5 References

[1] C. Gehrmann, H. Douglas, and D. K. Nilsson, “Are there good reasons for protecting mobile phones with hypervisors?” in IEEE Consumer Communications and Networking Conference, January 2011

[2] S. Chawla, A. Nigam, P. Doke, and Sanjay Kimbahune, "A Survey of Virtualization on Mobiles", In Proceedings of ACC (2)., pp. 430-441, 2011.

[3] C. Meng, Y. He, and Q. Zhang, "A Dependable, Scalable Maintenance Scheme for Stateful License in DRM", In Proceedings of Information Engineering and Electronic Commerce (IEEC '09), pp. 754 - 758, 2009.

[4] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, “seL4: formal verification of an OS kernel”, in J. N. Matthews and T. E. Anderson, editors SOSP, pages 207-220. ACM, 2009.

[5] E. Alkassar, M. A. Hillebrand, W. J. Paul, and E. Petrova, “Automated verification of a small hypervisor”, in VSTTE, volume 6217 of Lecture Notes in Computer Science, pages 40-54, Springer, 2010.

[6] C. Heitmeyer, M. Archer, E. Leonard, and J. McLean, “Applying formal methods to a certifiably secure software system”, in IEEE Trans. Softw. Eng., 34(1):82-98, January 2008.

[7] M. M. Wilding, D. A. Greve, R. J. Richards, and D. S. Hardin, “ Formal verification of partition management for the AAMP7G microprocessor”, Design and Verification of Microprocessor Systems for High-Assurance Applications, pages175-191, Springer US, 2010.

[8] Global Platform, "TEE Internal API Specification v1.0" , http://www.globalplatform.org/specificationsdevice.asp

[9] ARM Mali-V500, http://www.arm.com/products/multimedia/mali-video/mali-v500.php

[10] Mali-V500 Enabling HD and 4K everywhere, http://community.arm.com/servlet/JiveServlet/downloadBody/7791-102-3-12124/ATC329Slides_Androutsopoulos.pdf

[11] Intel Trusted Execution Technology http://www.intel.se/content/dam/www/public/us/en/documents/white-papers/trusted-execution-technology-security-paper.pdf

[12] R. Wojtczuk and J. Rutkowska., “Attacking Intel Trusted Execution Technology” http://invisiblethingslab.com/resources/bh09dc/Attacking%20Intel%20TXT%20-%20paper.pdf

[13] [Bluray Disc Playback on Intel Graphics FAQ
http://www.intel.com/support/graphics/sb/CS-029871.htm

[14] Understanding 8-channel LPCM over HDMI: Why it matters and Who supports it. http://www.anandtech.com/show/2622/2

[15] Intel Insider – What is it
http://blogs.intel.com/technology/2011/01/intel_insider_-_what_is_it_no/

[16] TV as an app
https://intel.activeevents.com/sf13/connect/fileDownload/session/A4860D8D1B3EF032DD8FEC080F9F1D46/SF13_EXPS005_100p.pdf

[17] Using Innovative Instructions to Create Trustworthy Software Solutions
http://software.intel.com/sites/default/files/article/413938/hasp-2013-innovative-instructions-for-trusted-solutions.pdf

[18] Output Content Protection and Windows Vista
https://www.google.se/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&ved=0CCsQFjAA&url=http%3A%2F%2Fdownload.microsoft.com%2Fdownload%2F5%2FD%2F6%2F5D6EAF2B-7DDF-476B-93DC-7CF0072878E6%2Foutput_protect.doc&ei=EeoNU83nI-iJ7AbQi4CQDA&usg=AFQjCNFyJoxWLkiWUbn0iyR-9z2-sU1a2w&sig2=D6dkmv04sGk6KirlX1dhMg&bvm=bv.61965928,d.ZGU

[19] Code signing for Protected Media Components in Windows Vista
http://msdn.microsoft.com/en-us/windows/hardware/gg487338.aspx

[20] ARM TrustZone Whitepaper
http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_TrustZone_security_whitepaper.pdf

[21] Designing with TrustZone Hardware Requirements
http://electronix.ru/forum/index.php?act=Attach&type=post&id=18827

[22] W. Jonker and J-P Linnartz, "Digital Rights Management in Consumer Electronics Products", IEEE Signal Processing Magazine, Vol. 21, No. 2, pp. 82-91, 2004.

[23] D. Halpin, " DRM and HTML5: it's now or never for the Open Web", the guardian, June 6.

[24] W3C draft, “Encrypted Media Extensions", https://dvcs.w3.org/hg/html-media/raw-file/tip/encrypted-media/encrypted-media.html, March, 2014.

[25] High-bandwidth Digital Content Protection System - Interface Independent Adaptation - 2.2, 2012., http://www.digital-cp.com/hdcp_

[26] GlobalPlatform TEE & ARM TrustZone technology: Building security into your platform

http://www.iaik.tugraz.at/content/about_iaik/events/ETISS_INTRUST_2013/ETISS/slides/GPTEE_Public.pdf

[27] R. Wang, Y. Shoshitaishvili, C. Kruegel, and G. Vigna, "Steal this movie - automatically bypassing DRM protection in streaming media services", USENIX Conference on Security, pp. 687-702, Washington D.C., USA, 2013

[28] S. Berger, R. Caceres, K. A. Goldman, R. Perez, R. Sailer, and L. Doorn, "vPTM: virtualizing the trusted platform module", USENIX Conference on Security, Vancouver, Canada, 2006

[29] J-L. Boulanger, "Industrial Use of Formal Methods: Formal Verification", Wily, 2012, ISBN: 9781118561829

[30] S. Michiels, W. Joosen, E. Truyen, and K. Versylpe, "Digital rights management – A survey of existing technologies", Department of Computer Science, Katholieke Universiteit Leuven, Tech. Rep., Nov. 2005.

[31] A.C. Prihandokoa, B. Litowa, and H. Ghodosia, "DRM rights protection capability - A review", International Conference on Computational Science and Information Management, pp. 12-17, Medan, Indonesia, 2012.

[32] Z. Zhang, " Digital rights management ecosystem and its usage controls: A survey", International Journal of Digital Content Technology and its Applications, Vol. 5, No. 3, pp. 255 - 272, 2011.

[33] J. Nützel., and A. Beyer. "How to increase the security of digital rights management systems without affecting consumer security". Emerging Trends in Information and Communication Security, pp. 368-380, 2006

[34] D. Deitcher. "Secure implementations of content protection (DRM) schemes on consumer electronic devices", ARM White Paper, 2013. http://arm.com/files/pdf/Secure_Implementation_of_Content_Protection_Schemes_on_Consumer_Electronic_Devices.pdf

[35] A. Vasudevan, E. Owusu, Z. Zhou, J. Newsome, and J. M. McCune, "Trustworthy execution on mobile devices: what security properties can my mobile platform give me?" Fifth International Conference on Trust and Trustworthy Computing, pp. 159-178), Vienna, Austria, 2012.

[36] J. Azema and G. Fayad, "M-Shield mobile security: making wireless secure", Texas Instruments White Paper, 2008, http://focus.ti.com/pdfs/wtbu/ti_mshield_whitepaper.pdf

[37] J. E. Ekberg, " Securing software architectures for trusted processor environment", PhD Thesis, Aalto University, 2013, http://lib.tkk.fi/Diss/2013/isbn9789526036328/isbn9789526036328.pdf

[38] H. Sundaresan, "OMAP platform security features", *Texas Instrument white paper*. 2003, http://focus.ti.com/pdfs/vf/wireless/platformsecuritywp.pdf

[39] S. G. Edward, C. Dwaine, G. Blaise, V. D. Marten, and D. Srinivas, “AEGIS: Architecture for Tamper-evident and Tamper-resistant Processing”, Seventeenth Annual International Conference on Supercomputing, pp. 160-171, San Francisco, USA. 2003.

[40] M. Dam, R. Guanciale, N. Khakpour, H. Nemati and O. Schwarz, "Formal Verification of Information Flow Security for a Simple ARM-Based Separation Kernel", to appear in the 20th ACM Conference on Computer and Communications Security, Berlin, November, 2013.

[41] R. Mijat and A. Nightingale, “Virtualization is coming to a platform near pou”, ARM White Paper, 2011, http://mobile.arm.com/files/pdf/System-MMU-Whitepaper-v8.0.pdf.

[42] Passera et al., “Protecting video content using virtualization”, USA Patent 8462945B2, http://www.google.com/patents/US8442224

[43] Lawler, Richard. “Intel confirms HDCP key is real, can now be broken at will”, http://arstechnica.com/tech-policy/2010/09/intel-confirms-the-hdcp-key-is-real-can-now-be-broken-at-will/

Abbreviations

AACS	Advanced Access Content System
AAMP7G	Advanced Architecture MicroProcessor 7 Government
AES-CTR	Advanced Encryption Standard Counter Mode
AKE	Authentication and Key Exchange
ASC	Address Space Controller
ASIC	Application Specific Integrated Circuit
AV	Audio/Video
BIOS	Basic Input/Output System
CEK	Content Encryption Key
CVP	Commercial Video Profile
CMLA	Content Management License Administrator
CPU	Central Processing Unit
DLNA	Digital Living Network Alliance
DMA	Direct Memory Access
DP	Display Port
DRM	Digital Rights Management
DRAM	Dynamic Random Access Memory
DTCP-IP	Digital Transmission Content Protection over IP
DVI	Digital Visual Interface
EAL7	Evaluation Assurance Level 7
GPU	Graphics Processing Unit
HD	High Definition
HDCP	High-Bandwidth Digital Copy Protection
HDMI	High Definition Multimedia Interface
HFS	Hardware Functionality Scan
HLOS	High Level Operating System
HYP	Hypervisor
KSV	Key Selection Vector
KTH	Kungliga Tekniska högskolan
MMU	Memory Management Unit
OEM	Original Equipment Manufacturer
OMA	Open Mobile Alliances
OMTP	Open Mobile Terminal Platform
REE	Rich Execution Environment
NS	Non-Secure Bus
NSAID	Non-Secure Access Identity Input
OS	Operating System
OVP	Open Virtual Platform
PAVP	Protected Audio Video Path
PE	Protected Environment
PTR	Private and Authenticated Tamper-Resistant
REE	Rich Environment Execution
RNG	Random Number Generator
ROM	Read Only Memory
RTP	Real Time Protocol
RTT	Round Trip Time
SoC	System on Chip
SCM	Secure Context Manager

SGX	Software Guard Extension
SICS	Swedish Institute of Computer Science
SMM	System Management Mode
SMMU	System Memory Management Unit
SKE	Session Key Exchange
SRTP	Secure Real Time Protocol
SMM	Secure State Machine
SSL	Secure Socket Layer
SVC	Secure Video Conferencing
TCG	Trusted Computing Group
TCB	Trusted Computing Base
TE	Tamper-Evident
TEE	Trusted Execution Environment
TZ	Trust Zone
TXT	Trusteed Execution Technology
TPM	Trusted Platform Module
TZC	Trust Zone Address Space Controller
vTPM	Virtual TPM
VCC	Verifying C Compiler
VM	Virtual Machine
VMM	Virtual Machine Manager
xPU	Protection Unit for peripheral ‘x’